UNITED STATES PATENT APPLICATION

PERIPHERAL DEVICE WITH HARDWARE LINKED LIST

INVENTOR
A. Kent Porterfield

Schwegman, Lundberg, Woessner, & Kluth, P.A.
1600 TCF Tower
121 South Eighth Street
Minneapolis, Minnesota 55402
ATTORNEY DOCKET 303.760US1
MICRON 01-0070

PERIPHERAL DEVICE WITH HARDWARE LINKED LIST

Technical Field of the Invention

The present invention relates generally to computer peripheral devices, and in particular, the present invention relates to hardware linked lists in computer peripheral devices.

Background of the Invention

Computers often have peripheral devices connected to them. Some peripheral devices are on cards that are plugged into slots, and other peripheral devices are located on the main circuit board within the computer. This main circuit board, which is typically called a "motherboard," usually has a processor attached.

Some peripheral devices can have one or more predefined capabilities that can be queried by the processor. For example, peripheral devices compliant with revision 2.2 of the peripheral component interconnect (PCI) local bus specification (published December 18, 1998 by PCI Special Interest Group, Portland, OR) may indicate one or more special predefined capabilities using a list of capabilities implemented in hardware and accessible by certain types of software. Examples of defined capabilities include power management, Advanced Graphics Processing (AGP), and PCI-X, which is a variation of PCI. In operation, software such as operating systems and device drivers may query a PCI peripheral device to determine if a particular capability is supported.

Capabilities lists are typically implemented as a set of defined groups of read-only registers. Each group of read-only registers for a given capability contains an address pointer to the next group of capabilities registers, forming what is commonly referred to as a linked list. The linked list is typically physically implemented as a read-only memory (ROM). These linked lists are generated during manufacture of the peripheral device, or during a one-time program operation, typically performed shortly after manufacture. As a result, the linked lists, and associated list of peripheral device

capabilities cannot be modified after the peripheral device is resident in a computer system.

For the reasons stated above, and for other reasons stated below which will become apparent to those skilled in the art upon reading and understanding the present specification, there is a need in the art for alternate hardware linked list implementations.

Summary of the Invention

The above mentioned problems and other problems are addressed by the present invention and will be understood by reading and studying the following specification.

In one embodiment, a peripheral device includes a hardware linked list that has a plurality of nodes. Each node includes a next node pointer register to point to the next node in the linked list. The peripheral device also includes a locking mechanism to conditionally make the next node pointer register of each node read-only.

In another embodiment, an integrated circuit includes a first writeable register to signify whether a capabilities list is enabled, and a second writeable register to point to a capabilities list. The integrated circuit also includes a write-once control register operable to make the first and second writeable registers read-only. The capabilities list can be a hardware linked list pointed to by the second writeable register. The hardware linked list includes a plurality of nodes, and each of the plurality of nodes includes a writeable next node register to point to the next node in the linked list. The writeable next node registers become read-only when the control register is written.

In another embodiment, a method of initializing a computer peripheral includes writing a list of capabilities to nodes in a hardware linked list within the computer peripheral and writing to a control register within the computer peripheral to make the nodes read-only. The nodes each include a capability register and a next node pointer register, and the next node pointer registers are modified when writing the list of capabilities.

Brief Description of the Drawings

Figure 1 is a perspective view of a computer system;

Figure 2 is a diagram of an integrated circuit with a hardware linked list;

Figures 3A, 3B, and 4 are diagrams of capabilities linked lists; and

Figure 5 is a flowchart of a method of initializing a computer peripheral.

Detailed Description of the Invention

In the following detailed description of the invention, reference is made to the accompanying drawings which form a part hereof, and in which is shown, by way of illustration, specific embodiments in which the invention may be practiced. In the drawings, like numerals describe substantially similar components throughout the several views. These embodiments are described in sufficient detail to enable those skilled in the art to practice the invention. Other embodiments may be utilized and structural, logical, and electrical changes may be made without departing from the scope of the present invention. The following detailed description is, therefore, not to be taken in a limiting sense, and the scope of the present invention is defined only by the appended claims, along with the full scope of equivalents to which such claims are entitled.

Figure 1 is a perspective view of a computer system. Computer system 100 includes motherboard 102, processor 104, basic input/output software (BIOS) 130, and peripherals 108, 110, 120 and 122. Processor 104, BIOS 130, and peripherals 108 and 110 reside directly on motherboard 102. Peripherals 120 and 122 in contrast, are add-in cards accepted by connectors 112 and 114 mounted to motherboard 102. Processor 104, BIOS 130, and peripherals 108, 110, 120, and 122 are coupled together by bus 106.

Processor 104 can be any type of processor. For example, in some embodiments processor 104 is a microprocessor. Also for example, in some embodiments processor 104 is a digital signal processor or a microcontroller. BIOS 130 is a memory device that includes instructions for initialization of computer system 100, and also includes instructions for low-level software routines. For example, when power is applied to

computer system 100, processor 104 reads initialization instructions from BIOS 130 and performs various initialization tasks. Initialization tasks performed by processor 104 include internal initialization of processor 104 and initialization of peripherals, such as peripherals 108, 110, 120, and 122. Processor 104 communicates with BIOS 130 and various peripherals using bus 106. In some embodiments, bus 106 is a communications bus that adheres to a standard protocol, such as PCI local bus specification, revision 2.2.

Initialization tasks performed by processor 104 include initialization of peripherals such as peripherals 108, 110, 120, and 122. For example, a capabilities list held within peripherals can be initialized by processor 104. Example embodiments of hardware linked lists useful for defining lists of peripherals capabilities are explained with reference to the figures that follow.

Figure 2 is a diagram of an integrated circuit with a hardware linked list. Integrated circuit 200 is an integrated circuit that includes a linked list in hardware. Integrated circuit 200 can be any integrated circuit that benefits from a hardware linked list. For example, in some embodiments, peripheral devices 108 and 110 (Figure 1) include the linked list circuitry shown in integrated circuit 200. Also for example, peripherals 120 and 122 (Figure 1) can include integrated circuits such as integrated circuit 200.

Integrated circuit 200 includes processor interface 250, control register 240, list enabled register 202, head pointer register 204, and list nodes 210, 220, and 230. Each of list nodes 210, 220, and 230 include a next node pointer register and one or more linked list information registers. For example, list nodes 210, 220, and 230 include next node pointer registers 212, 222, and 232, and also include linked list information registers 214, 224, and 234, respectively. In general, list nodes are groups of registers and each next node pointer register points to the next group.

List nodes within integrated circuit 200 can be logically chained to create a linked list in hardware. For example, each next node pointer register can include the address of a list node, thereby providing a logical chain between list nodes. Logical chaining of list nodes is shown in more detail in the figures below.

Linked list information registers include information useful to integrated circuit 200. In some embodiments, linked list information registers include information regarding capabilities of integrated circuit 200. When a linked list is created by logically chained list nodes, each linked list information register includes a subset of the information held by the entire linked list. Three list nodes are shown in Figure 2. In some embodiments, many more than three list nodes exist. In other embodiments, only one linked list node exists. In the examples that follow, three list nodes are used repeatedly for clarity, and for ease of comparison between the various figures.

Head pointer register 204 is a register within integrated circuit 200 that provides a starting address for the hardware linked list. For example, head pointer register 204 can include the address of any of the list nodes 210, 220, or 230. Head pointer register 204 is said to "point" to the list node corresponding to the address held in head pointer register 204. List enabled register 202 is a register within integrated circuit 200 that holds information regarding the validity of the hardware linked list. In some embodiments, list enabled register 202 is a single bit in a larger status register. In other embodiments, list enabled register is a stand alone register with a dedicated address.

In embodiments represented by Figure 2, processor interface 250 communicates with a bus that includes data, address, and control. For example, processor interface 250 can be coupled to a bus such as bus 106 (Figure 1). One skilled in the art will understand that many different types of busses exist, and a processor interface 250 can be readily made to communicate with any type of bus. Processor interface 250 allows other integrated circuits, such as a processor, to communicate with the various registers shown in Figure 2. Each of the registers shown in integrated circuit 200 have a memory mapped address associated therewith, and processor interface 250 provides a mechanism for the various registers within integrated circuit 200 to be accessed at their respective addresses. For example, using internal node 252, processor interface 250 can write to, and read from, list enabled register 202, head pointer register 204, and the various registers within list nodes 210, 220, and 230. Processor interface 250 can also write to and read from control register 240 using internal node 252. Internal node 252 is

shown in Figure 2 as a single line, but in many embodiments, internal node 252 includes many physical lines for decoding and data transfer.

Control register 240 receives a system reset on node 244, and produces a "read-only" signal on node 242. When power is applied to integrated circuit 200, or when a hardware reset is asserted, reset signal on node 244 is asserted to control register 240. Control register 240, in turn, de-asserts the read-only signal on node 242. When the read-only signal on node 242 is de-asserted, the various registers within integrated circuit 200 are not read-only. For example, when the read-only signal on node 242 is de-asserted, processor interface 250 can read and write to list enabled register 202, head pointer register 204, and the various registers of list nodes 210, 220, and 230.

Control register 240 can also assert the read-only signal on node 242, thereby removing the ability to write to the various registers of integrated circuit 200. In some embodiments, when control register 240 is written to by processor interface 250, the read-only signal on node 242 is asserted. Subsequent to the read-only signal being asserted on node 242, the various registers within integrated circuit 200 are no longer writeable, but instead are read-only. In some embodiments, control register 240, once written to, cannot be written to again prior to a reset signal being asserted on node 244.

In operation, after a system reset is asserted on node 244, various registers within integrated circuit 200 are writeable. In some embodiments, a linked list can be created by writing to the various registers within integrated circuit 200. For example, list enabled register 202 can be modified to indicate that a linked list is enabled within integrated circuit 200, and head pointer register 204 can be modified to include the memory mapped address of any of list nodes 210, 220, and 230. Furthermore, each next node pointer register can be modified in the same manner as head pointer register 204 to point to subsequent list nodes.

In some embodiments, in addition to head pointer register 204 and the next node pointer registers being modified, the linked list information registers can also be modified. For example, list nodes intended to be included within the hardware linked list can have various information written thereto, allowing the hardware linked list to

provide various types of information. Once the hardware linked list is created by modifying registers as previously described, control register 240 can be written to, thereby asserting the read-only signal on node 242 rendering the various registers within integrated circuit 200 read-only.

The sequence just described can be useful to allow initialization software to create a linked list within integrated circuit 200, and then lock the list such that it cannot be modified prior to a subsequent system reset. For example, referring now back to Figure 1, BIOS 130 can include processor instructions for initialization of peripherals 108, 110, 120, and 122. Each of these peripherals can include the circuitry represented in Figure 2, and processor 104 can generate a linked list by modifying the various register values prior to writing to the control register.

As previously described, control register 240 provides a read-only signal on node 242 to registers internal to integrated circuit 200. In other embodiments, a read-only signal is provided to registers using mechanisms other than control register 240. For example, a read-only signal can be brought in from a pin on integrated circuit 200, thereby allowing an external signal to control the read-only capability of the various registers within integrated circuit 200.

The circuitry shown in Figure 2 is useful in any integrated circuit that can benefit from a writeable hardware linked list that can be made read-only. For example, in a PCI local bus compliant peripheral, a capabilities linked list can be generated by low-level software and made read-only to high-level software. Examples of such embodiments are shown in the figures that follow.

Figure 3A shows a PCI local bus compliant capabilities list 300. Capabilities list 300 includes capabilities list enabled register 302, capabilities pointer register 304, capabilities lock register 340, and capabilities list nodes 310, 320, and 330. Capabilities list enabled register 302 corresponds to list enabled register 202 (Figure 2). Likewise, capabilities pointer register 304 corresponds to head pointer register 204 (Figure 2). In a similar manner, list nodes 310, 320, and 330 correspond to list nodes shown in Figure

2. Each list node includes a capability register and a next capability register. For example, list node 310 includes capability register 314 and next capability register 312.

Linked list 300 is shown in Figure 3A having already been programmed. For example, capabilities pointer register 304 includes address A, which is the memory mapped address of linked list node 310. Similarly, the next capabilities registers of list nodes 310 and 320 hold addresses B and C, which correspond to memory mapped addresses of list nodes that logically follow. List node 330 is at the end of linked list 300, and the next capability register 332 holds a null value, representing the end of the linked list. Linked list 300 also includes capabilities lock register 340. Capabilities lock register 340 corresponds to control register 240 (Figure 2). When capabilities lock register 340 is written to, the read-only signal on node 342 is asserted, thereby making the various registers in linked list 300 read-only.

Linked list 300 includes three linked list nodes representing capabilities A, B, and C. When linked list 300 is traversed, capability registers 314, 324, and 334 will be found within linked list 300, corresponding to capabilities A, B, and C. For example, referring now back to Figure 1, when linked list 300 is included in a peripheral such as peripheral 108, processor 104 can query the peripheral to ascertain capabilities by traversing linked list 300.

Figure 3B shows a hardware linked list utilizing the same list nodes as Figure 3A, but having a different logically linked list. Linked list 360 includes capabilities list enabled register 302, capabilities pointer register 304, capabilities lock register 340, and list nodes 310 and 330. Prior to read-only signal on node 342 being asserted, the various registers in linked list 360 were programmed to include list nodes 310 and 330, and exclude list node 320. When linked list 360 is traversed, only list nodes corresponding to capabilities A and C will be found. After capabilities lock register 340 is written to, linked list 360 will be read-only, and list node 320 will not be found.

Figures 3A and 3B show two of many possible embodiments that can be generated by modifying registers within the linked list. As shown in Figures 3A and 3B, possible capabilities include A, B, and C. In the embodiment of Figure 3A, all three

capabilities are included in the linked list in the order A, B, and C. In the embodiment of Figure 3B, only capabilities A and C are included. Any number of list nodes can be included in a linked list. For example, in one embodiment, only list node 320 is included, and in another embodiment only list node 310 is included. Furthermore, list nodes can be included in any order. For example, capabilities pointer 304 can hold the memory mapped address of any list node included within the linked list. Additionally, any next capability register can include the memory mapped address of any list node desired.

In some embodiments, list nodes with predefined capabilities registers exist within peripheral devices prior to power being applied. For example, a peripheral device that includes list nodes 310, 320, and 330 can have capabilities registers 314, 324, and 334 preprogrammed with capabilities identifiers for capabilities A, B, and C, respectively. Low-level software then builds a linked list by simply modifying the list head pointer and next capability pointers. In other embodiments, capabilities identifiers are not preprogrammed within list nodes, and linked lists can be built by modifying capabilities registers and pointer registers.

Figure 4 shows another embodiment of a hardware linked list in an integrated circuit. Integrated circuit 400 includes head pointer register 404 and list node groups 402 and 450. List node group 402 includes list nodes 410, 420, and 430, and list node group 450 includes list nodes 460, 478, and 480. In the embodiment shown in Figure 4, head pointer register 404 can be modified to select one of two groups of list nodes. Head pointer register 404 is shown pointing to the linked list that includes group 402. A dashed line is shown from head pointer register 404 to list node 460 to signify the capability of changing head pointer register 404 to point to list node 460. Once head pointer register 404 is modified to point to either group 402 or 450, control register 490 is written to render all registers read-only, as in the previously described embodiments.

Figure 5 shows a flowchart of a method for initializing a peripheral. Method 500 begins at 510 where instructions are read from a memory device that holds basic input/output software. For example, processor 104 (Figure 1) can read initialization

instructions from BIOS 130 (Figure 1). At 520, a hardware linked list is enabled by writing to a list enabled register. This corresponds to processor 104 writing to a list enabled register such as register 202 (Figure 2), or register 302 (Figure 3A). At 530, a list head pointer register is written. This corresponds to a head pointer register such as register 204 (Figure 2) or register 304 (Figure 3A). At 540, a link within the hardware linked list is modified by writing a next node pointer register. The next node pointer register of method 500 corresponds to next capability registers 312, 322, and 332 of Figures 3A and 3B, and also corresponds to next node pointer registers 212, 222, and 232 of Figure 2. By writing to next node pointer registers, links within the hardware linked list are modified to either include or exclude list nodes within the hardware linked list. At 550, a control register is written to render the list read-only.

One skilled in the art will understand that many embodiments exist and that specific examples have been described. For example, in some embodiments only pointer registers are writeable and capabilities registers are always read-only. In other embodiments, pointer registers and capabilities registers are writeable prior to a write to a control register which renders them all read-only. Any combination of writeable registers and read-only registers can be included without departing from the scope of the present invention.

Although specific embodiments have been illustrated and described herein, it will be appreciated by those of ordinary skill in the art that any arrangement which is calculated to achieve the same purpose may be substituted for the specific embodiment shown. This application is intended to cover any adaptations or variations of the present invention. Therefore, it is manifestly intended that this invention be limited only by the claims and the equivalents thereof.